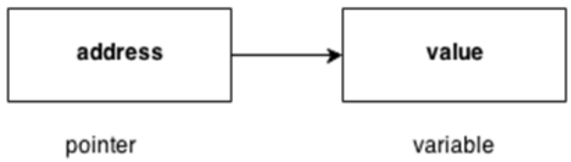
Object-Oriented Programming I (CSE 121)
Week # 04 (YouTube Lecture: 07 & 08)
Pointers in C++

Objective(s)

In this chapter, you will learn:

- Pointers definition, syntax, examples and benefits

A **pointer** is a variable that holds the address of another variable. A **pointer** however, is a variable that stores the memory address as its value. Pointers are used to store addresses rather than values.



Syntax of pointer

Here is how we can declare pointers.

data_type *pointer_name;   Example: int *pc;

We can also declare pointers in the following way.

data_type* pointer_name;   Example: int* pc; //preferred syntax

Example of pointer

In this example we print the value that **pc** points to.

```
pointer.cpp
#include <iostream>
using namespace std;

int main(){
    int c = 22;
    int *pc;
    pc = &c;
    cout << "Value at pc = " << pc << endl;
    cout << "Value at c = " << c << endl;
    cout << "Value at *pc = " << *pc << endl;
}
```

```
C:\Users\Bulbul\Desktop\poir
Value at pc = 0x61fe14
Value at c = 22
Value at *pc = 22
```

In the example above we used **ampersand** sign (&). This sign is called the reference operator. If the reference operator is used you will get the "address of" a variable.

Note: If you forget to place * (in front of the pointer) in the **cout** statement, you will print the address of integer **c**.

Symbols used in pointer

Symbol	Name	Description
& (ampersand sign)	Reference/address operator	Determine the address of a variable.
* (asterisk sign)	Dereference/indirection operator	Access the value of an address.

Note: the asterisk operator * has two semantic meanings: to declare a pointer variable (i.e. int *pc) and also to access the value a pointer points to (*pc = 22).

We can say:

- & is the reference operator and can be read as "address of".
- * is the dereference operator and can be read as "value pointed by".

Using pointers to pass values

In the example above we used an integer '**c**' of which we stored the "address of" into a pointer '**pc**'. But it is also possible to use a pointer to pass a value to a variable.

```
#include<iostream>
using namespace std;

int main (){
    int c;
    int* pc;
        pc = &c;
        *pc = 22;
    cout << "Value at c = " << c << endl;
    return 0;
}
```

```
C:\Users\Bulbul\De
Value at c = 22
```

Note: You can see that no value is stored in **c** as in the previous example (**c** = 22). But the result is the same.

First, we store the address of **c** into the pointer ***pc**. Then we say we want the value 22 stored at the address where the pointer is pointing to (in this case **c**).

Benefits of pointer

- We can return multiple values from function using pointer.
- It makes you able to access any memory location in the computer's memory.
- Pointer reduces the code and improves the performance, it is used to retrieving strings, trees etc. and used with arrays, structures and functions.

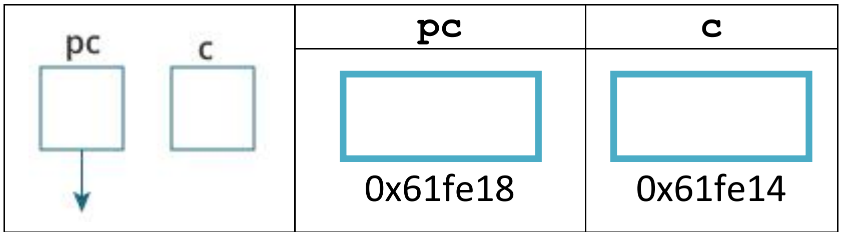
Another Example

```
#include<iostream>
using namespace std;

int main(){
    int *pc,c;
    cout << "Address of pointer variable pc: " << &pc << endl;
    cout << "Address of normal variable c: " << &c << endl << endl;
        c = 22;
    cout << "Value of normal variable c: " << c << endl << endl;
        pc = &c;
    cout << "Address stored at pointer variable pc: " << pc << endl;
    cout << "Address of normal variable c: " << &c << endl;
    cout << "Address of pointer variable pc: " << &pc << endl;
    cout << "Value of normal variable c: " << c << endl;
    cout << "Value of pointer variable pc(*pc): " << *pc << endl << endl;//value of c
        c = 11;
    cout << "Value of normal variable c: " << c << endl;
    cout << "Value of pointer variable pc(*pc): " << *pc << endl << endl;//value of c
        *pc = 5;
    cout << "Value of normal variable c: " << c << endl;
    cout << "Value of pointer variable pc: " << *pc << endl;//value of c
    return 0;
}
```

Explanation of the program

1. `int* pc, c;`



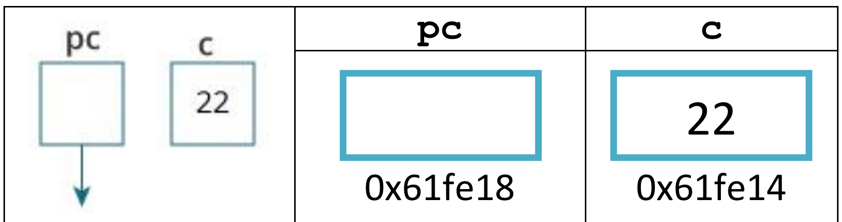
Variable Name	Value	Address
pc		0x61fe18
c		0x61fe14

Here, ***pc*** is a pointer variable, and ***c*** is a normal variable, both variable types are **int**.

Since ***pc*** and ***c*** are not initialized at first, pointer ***pc*** points to either no address or a random address. And, variable ***c*** has an address but contains a random garbage value.

Note: You may not get the same results when you run the program.

2. `c = 22;`

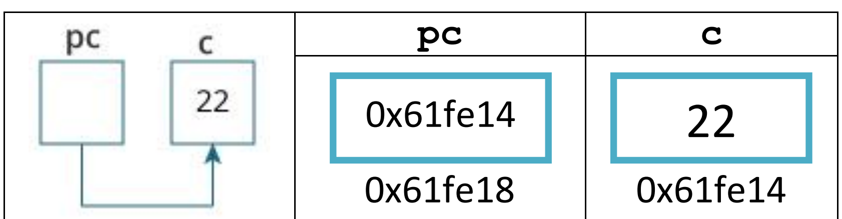


Variable Name	Value	Address
pc		0x61fe18
c	22	0x61fe14

This assigns 22 to the variable ***c***, i.e., 22 is stored in the memory location of variable *c*.

Note: Address is expressed as a hexadecimal number.

3. `pc = &c;`

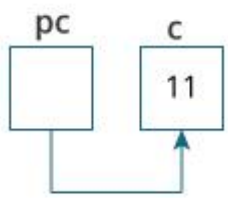


Variable Name	Value	Address
pc	0x61fe14 (address of c)	0x61fe18
c	22	0x61fe14

This assigns the address of variable ***c*** to the pointer ***pc***.

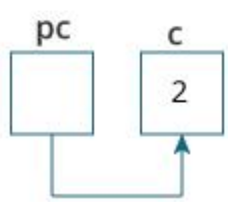
You see the value of ***pc*** is same as the address of ***c*** and the content of ***pc*** is 22 as well.

4. `c = 11;`

	**pc**	**c**
	0x61fe14 0x61fe18	11 0x61fe14
Variable Name	**Value**	**Address**
pc	0x61fe14	0x61fe18
c	11	0x61fe14

This assigns 11 to variable ***c***. Since, pointer ***pc*** points to the same address as ***c***, value pointed by pointer ***pc*** is 11 as well.

5. `*pc = 2;`

	**pc**	**c**
	0x61fe14 0x61fe18	2 0x61fe14
Variable Name	**Value**	**Address**
pc	0x61fe14	0x61fe18
c	2	0x61fe14

This change the value at the memory location pointed by pointer ***pc*** to 2.
Since the address of the pointer ***pc*** is same as the address of ***c***, value of ***c*** is also changed to 2.

Common mistakes when working with pointers

Suppose, we want pointer **pc** to point to the address of **c**. Then,

int c, *pc;

pc = c; // **Wrong! pc** is an address whereas, **c** is not an address.

***pc = &c;** // **Wrong!** &c is an address whereas, ***pc** is the value stored in **&c**

pc = &c; // **Correct! pc** is an address and **&c** is also an address.

***pc = c;** // **Correct!** Both ***pc** and **c** are values

Program Examples

1. Write a C++ program to add two numbers using pointers.

example.cpp

```cpp
#include<iostream>
using namespace std;

int main(){
    int a, b, sum, *pa, *pb;
    cout << "Enter the value of a & b = ";
    cin >> a >> b;
        pa = &a;
        pb = &b;
        sum = *pa + *pb;
    cout << "a + b = " << sum << endl;
    return 0;
}
```

C:\Users\Bulbul\Documents\example.exe

```
Enter the value of a & b = 10 15
a + b = 25
```

2. Write a C++ program to swap two numbers using pointers.

example.cpp

```cpp
#include<iostream>
using namespace std;

int main()
{
    int a = 10, b = 20, *p1, *p2, temp;
    p1 = &a;
    p2 = &b;
    cout << "Before Swapping: ";
    cout << "a = " << a << "\tb = " << b << endl;
        temp = *p1;
        *p1 = *p2;
        *p2 = temp;
    cout << "After Swapping: ";
    cout << "a = " << a << "\tb = " << b << endl;
    return 0;
}
```

C:\Users\Bulbul\Documents\example.exe

```
Before Swapping: a = 10 b = 20
After Swapping: a = 20  b = 10
```

3. Write a C++ program to find the square of a number using pointer.

example.cpp

```cpp
#include<iostream>
#include<cmath>
using namespace std;

int main(){
    int n, *pn;
    cout << "Enter the value of n: ";
    cin >> n;
        pn = &n;
    cout << "Using pointer: " << pow(*pn,2) << endl;
    cout << "Without pointer: " << pow(n,2) << endl;
    return 0;
}
```

C:\Users\Bulbul\Documents\ex

```
Enter the value of n: 7
Using pointer: 49
Without pointer: 49
```

…………………